Table of Contents

File Transfers

File Transfer: Overview

Here's a general overview of the various file transfer scenarios within the NAS environment, with pointers to related articles.

File Transfers Between Pleiades, Columbia, and Lou

File transferring between NAS systems in the secure enclave (Pleiades, Columbia, and Lou) uses host-based authentication (transparent to users) and is usually straightforward. The following articles provide basic information to help you get started.

- Local File Transfer Commands - cp, cxfscp
- Remote File Transfer Commands - scp, bbftp/bbscp
- File Transfer Between Pleiades and Columbia or Lou
- Transferring Files from the Pleiades Compute Nodes to Lou
- Checking File Integrity

File Transfers between a NAS HECC Host and Your Localhost

Transferring files between a NAS host (such as Pleiades,Columbia, or Lou) and a remote host, such as your local desktop, is more complex. There are multiple factors that you should be aware of:

Which commands to use

Remote File Transfer Commands such as *scp* and *bbftp* and *bbscp* are supported on most NAS high-end computing systems. Depending on the way the transfers are performed, you may need either one or both of the client and server software of scp and/or bbftp or the bbscp script installed on your localhost.

- **Transfer Rate**

  File transfer rate with scp, especially using scp from versions of Open that SSH are older than 4.7, can be as slow as 2 MB/sec. For transferring large files over a long distance, consider the following:

  - upgrade to the the latest version of OpenSSH
  - apply the HPN-SSH patch to your OpenSSH

    - enable compression by adding -C to the scp command-line if the data will compress well
    - use bbftp/bbscp

- **Security Issues**
    - With scp, users' authentication information (such as password or passcode) and data are encrypted.
    - With bbftp and bbscp, only the authentication information is encrypted, while data is not.
    - You can use GPG to encrypt your data prior to the transfer.

Where transfer commands are initiated

- **Outbound file transfers**

  When the file transfer command is initiated on a NAS host such as Pleiades, Columbia, or Lou, the transfer does not need to go through SFE[1,2] or Secure Unattended Proxy. This is the easiest way to transfer files from and/or to your site if your localhost is configured to allow the transfer.

  To learn more, see also Outbound File Transfer Examples.

- **Inbound file transfers**

  When the file transfer command is initiated on a remote host such as your local desktop, the transfer must go through either SFE[1,2] or Secure Unattended Proxy.

    - *Going through SFE[1,2]*

      Going through SFE[1,2] requires authentication via your RSA SecurID fob at the time of operation; you will be prompted for your passcode when you issue the file transfer commands, such as *scp*, *bbftp*, or *bbscp*.

      Transfers can be done with one of the following two approaches:

        1. Two steps: Initiate scp from your localhost to SFE[1,2], and then initiate another scp from SFE[1,2] to Columbia, Pleiades or Lou.

           WARNING: Do not store files on the SFEs since space is very limited. Any file transfers though the SFE really should use the SSH pass-through option described next.
        2. One step: Initiate scp, bbftp/bbscp from your localhost to PleiadesmColumbia, or Lou if SSH Passthrough has been set up.

      To learn more, see also Inbound File Transfers through SFEs Examples.

♦ *Going through SUP*

Going through the Secure Unattended Proxy does *not* require SecurID fob authentication at the time of operation. Instead, special "SUP keys" using SecurID authentication must be obtained ahead of time. The "SUP keys" are good for one week and are used automatically to authenticate users for file transfers using scp, bbftp or bbscp issued on a command-line or in a job script.

WARNING: Although users have accounts on the SUP servers, no login session is allowed.

File transfers going through SUP offers multiple benefits over going through the SFEs:

◊ SUP allows the transfer to be unattended; that is, you do not have to type in your password, passphrase, or passcode when the file transfer command is issued. So, file transfers can be done within a script that can be scheduled to run ahead of time. On the other hand, file transfers through the SFEs can not be done in a script.
◊ File transfers through SUP are done in one step, and setting up SSH passthrough is not needed since the SFEs are not involved.
◊ SUP automatically sets some options, such as the port range allowed for bbftp transfers, so that you don't have to set them explicitly. Thus, the syntax for bbftp over SUP is greatly simplified compared to bbftp without SUP.

NOTE: Some sites only allow specific outbound ports; this may cause bbftp to break.
◊

See the article Using the Secure Unattended Proxy (SUP) and the examples there for more information.

• **File staging**

When there are issues (such as a firewall) that hinder the inbound and/or outbound transfers, file staging through DMZFS[1,2] is another option. You can deposit and retrieve files on DMZFS[1,2] by issuing the *scp* or *bbftp* command on either a NAS host or your localhost.

WARNING: The total storage space on DMZs is 2.5TB, shared among all users; files older than 24 hours are removed.

DMZFS[1,2] do not use SecurID fob for authentication. Instead, password or public key authentication is used for file transfers via DMZFS[1,2].

Unattended file transfers can also be done through DMZFS[1,2] if public key authentication has been set up on DMZFS[1,2].

Note, however, that for this purpose, the SUP is preferred as SUP transfers are direct to the end target so do not have the storage restrictions and two step performance limitations of DMZFS when using bbftp/bbscp.

Read File Staging through DMZ File Servers for more information.

NAS Username and Your Local Username

If your NAS username and local username are different, you may have to add the appropriate username in the scp, bbftp or bbscp command-line.

- If you issue the command on your local host, then the username is your NAS username.
- If you issue the command on a NAS host, then the username is your local username.

In the examples shown in the articles Outbound File Transfer Examples and Inbound File Transfers through SFEs Examples, you will find the correct syntax for adding the appropriate username in the file transfer commands.

For inbound file transfers, if you have correctly included your NAS username in the *~/.ssh/config* file of your localhost, you do not have to include the NAS username in the *scp*, *bbftp* or *bbscp* command. A template for the *~/.ssh/config* is available for download.

Check File Integrity Before and After the Transfer

It's a good practice to ensure the integrity of the data before and after the transfer. For more information, see Checking File Integrity.

Tuning your Local System to Improve File Transfer Performance

Some file transfer commands provide options that can be used to improve your transfer rates. For example, enabling compression during file transfers may help in some cases: with bbftp, you can use multiple streams instead of a single stream for better performance. Read Tips for File Transfers for more information.

On the other hand, file transfer performance is also dependent on some system-wide settings. If necessary, ask your local system administrator to look into issues discussed in the following articles:

- TCP Performance Tuning for WAN Transfers
- Optional Advanced Tuning for Linux
- Pittsburgh Supercomputing Center's Enabling High Performance Data Transfers - a properly tuned TCP/IP stack

Local File Transfer Commands

DRAFT

This article is being reviewed for completeness and technical accuracy.

The following file transfer commands can be used when both the source and destination locations are accessible on the same host where the command is issued. Basic information about each command is provided below.

- **cp**:

  cp is a UNIX command for copying files between two locations (for example, two different directories of the same filesystem or two different filesystems such as NFS, CXFS or Lustre).

  - *Where is it installed at NAS?*

    cp is available on all NAS systems except SFE[1,2], DMZFS[1,2], Bouncer and Bruiser.

  - *Examples:*

    ```
    pfe1% cp $HOME/foo $HOME/newdir/foo2
    pfe1% cp $HOME/foo /nobackup/username
    ```

- **cxfscp**:

  cxfscp is a program from SGI for quickly copying large files to and from a CXFS filesystem (for shared-memory systems such as Columbia). It can be significantly faster than cp on CXFS filesystems since it uses multiple threads and large direct I/Os to fully utilize the bandwidth to the storage hardware.

  For files less than 64 kilobytes in size, which will not benefit from large direct I/Os, cxfscp will use a separate thread for copying these files using buffered I/O similar to cp.

  - *Where is it installed at NAS?*

    cxfscp is installed on cfe2, c21-24, Lou[1-2], and the Pleiades bridge nodes (bridge[1-2]). It is not available on the Pleiades front-end nodes (pfe[1-12]).

  - *When to use it?*

    The Columbia CXFS filesystems (/nobackup[1-2][a-i]) are mounted on all Columbia hosts (cfe2, c21-24), Lou[1-3], and the Pleiades bridge nodes (bridge[1,2]). The command cxfscp can be issued on any of these hosts to copy large files to and from Columbia's /nobackup[1-2][a-i]. This is an easy way to transfer files among Columbia, Pleiades and Lou without the need for scp, bbftp or bbscp.

- *Examples:*

```
cfe2% cxfscp /nobackup2a/username/foo /nobackup2a/username/new_dir
lou2% cxfscp /nobackup2a/username/foo $HOME
bridge2% cxfscp $HOME/foo /nobackup2a/username
bridge2% cxfscp /nobackupp20/username/foo /nobackup2a/username
```

- *Performance:*

  Some benchmarks done by NAS staff show that cxfscp is typically 4 - 7 times faster than cp for large files (2+ GB) and can achieve upto 400 MBytes/sec.

For more information, read **man cxfscp**.

Remote File Transfer Commands

DRAFT

This article is being reviewed for completeness and technical accuracy.

The following file transfer commands can be used when the source and destination are located at different hosts. They can be used to transfer files either between NAS HECC hosts or between a NAS host and a remote host such as your local desktop system.

- **scp (with/without HPN-SSH patch)**

  Secure Copy Protocol (SCP), based on Secure Shell Protocol (SSH), is a means of securely transferring files between a local and a remote host. Both the authentication information (such as password or passcode) and user's data are encrypted.

  Normal scp (without the HPN-SSH patch)

  The most widely used scp is from OpenSSH.

  - *Where is it installed at NAS?*

    A copy of scp from OpenSSH without the patch is available on the Pleiades front-end and bridge nodes (pfe[1-12], bridge[1-2]), all Columbia nodes, Lou[1-2], SFE[1,2], Bouncer, and Bruiser.

    The copy on SUP contains the HPN-SSH patch.

    scp is not available on DMZFS[1,2]. Use scp on Columbia, Pleiades, Lou or your localhost to push files into DMZFS[1,2] or pull files out of DMZFS[1,2].

  - *Do you need it installed on your localhost?*

    If you already have a version of SSH installed on your localhost, most likely, scp is already there.

  - *When to use it?*

    scp is typically used for transferring small files (<< 5GB) within NAS or offsite (<< 1 GB) that takes reasonable amount of time to complete.

  - *Examples:*

    For outbound transfer:

lou1% scp local_username@your_localhost.domain:foo ./foo2

For inbound two-step transfer:

```
your_localhost% scp foo nas_username@sfe1.nas.nasa.gov:foo2
sfe1% scp foo2 lou1:
```

For inbound one-step transfer if SSH-passthrough has been set up correctly:

your_localhost% scp foo nas_username@lou1.nas.nasa.gov:foo2

To transfer files through DMZFS[1,2], initiate the scp command from either a NAS HECC host or your localhost, not DMZFS[1,2]. For example,

```
your_localhost% scp foo nas_username@dmzfs1.nas.nasa.gov:foo2
pfe1% scp dmzfs1:foo2 .
```

Omit *local_username@* and *nas_username@* in the examples above if your local username and NAS username are identical.

- *Performance:*
    - ◊ Within NAS HECC Enclave, depending on source and destination hosts and other factors, the performance range will be 40 - 100 Mbytes/sec.
    - ◊ Over WAN (such as between NAS and a remote site), the best you get with scp from OpenSSH versions older than 4.7 (with the internal channel buffer set to 128 KB) is ~ 2 MBytes/sec. Starting with OpenSSH version 4.7, a larger channel buffer is introduced to improve file transfer performance. Users are recommended to upgrade to version 4.7 or later.

      In case where OpenSSH 4.7 or a later version does not yield satisfactory performance, consider applying the HPN-SSH patch to your OpenSSH.

      If the data you are transferring will compress well, consider enabling compression by adding -C to your scp command-line.

HPN-SSH enabled scp

HPN-SSH is a patch for OpenSSH designed to eliminate a network throughput bottleneck that typically occurs in an SSH session over long distance and high bandwidth network (i.e.,when the bandwidth-delay product is high). This is accomplished by allowing internal flow control buffers to be defined and grow at runtime, rather than statically defining them as OpenSSH does. The resulting performance increase can range from 10x to more than 50x, depending on the cipher used and host tuning.

HPN-enabled SSH is fully interoperable with other SSH servers and clients. HPN clients will be able to download faster from non-HPN servers, and HPN servers will be able to receive uploads faster from non-HPN clients. However, the host receiving the data must have a properly tuned TCP/IP stack.

Ask your local network staff for help to see if an HPN-SSH patch is needed for certain network connection.

- *Where is it installed at NAS?*

  - ◊ On cfe2, the client version of OpenSSH 4.7p1 with HPN12v20 patch is available.
  - ◊ On Lou[1-2], the client version of OpenSSH 5.0p1 with HPN13v1 patch is available.
  - ◊ On SUP, both the client and server of OpenSSH 5.1p1 have been patched with HPN13v5.

  On cfe2 and Lou[1-2], the HPN-patched SSH programs are purposely named as *hpn-ssh, hpn-scp,* and *hpn-sftp* to distinguish them from the default non-HPN versions (*ssh, scp* and *sftp*) of OpenSSH.

- *Do you need it installed on your localhost?*

  To get good performance, an HPN-SSH server must be installed on your local system if data is to be received on your local system.

  Typical installation procedure:

  1. Download OpenSSH source (*openssh-x.xpx.tar.gz*) from http://www.openssh.com
  2. Download corresponding HPN SSH patch (*openssh-x.xpx-hpnxxvx.diff.gz*) from http://www.psc.edu/networking/projects/hpn-ssh
  3. Uncompress and untar above distributions
  4. move the file *openssh-x.xpx-hpnxxvx.diff* to the directory *openssh-x.xpx*
  5. cd openssh-x.xpx (for example, openssh-5.0p1)
  6. patch < openssh-5.0p1-hpn.diff
  7. configure [OPTIONS]
  8. make [OPTIONS]
  9. Validate:
     ```
     %ssh -v
     OpenSSH_5.0p1-hpn13v3
     ```

- *Examples:*

```
lou[1-2]% hpn-scp -c arcfour -o TCPRcvBufPoll=yes source destination
your_localhost% scp -c arcfour -o TCPRcvBufPoll=yes source destination
```

Note:

- ◊ arcfour (RC4) is a more CPU-efficient 128-bit cipher. One can also choose NONE for cipher so that there is no encryption for data.
- ◊ Enabling *TCPRcvBufPoll* allows for the TCP receive buffer to be polled through the duration of the connection.

♦ *Performance:*

With an HPN-SSH enabled scp, one can expect good performance for transferring large files to remote sites over long distance with high latency connections. Improvement over non-patched scp older than 4.7 (2 Mbytes/sec) may be 10x to 50x.

- **bbFTP**

bbFTP is a high performance remote file transfer protocol which supports parallel TCP streams for data transfers. Basically, it splits a single file in several pieces and sends them through parallel streams. The whole file is then rebuilt on the remote site. bbFTP also allows dynamically adjustable TCP/IP window sizes instead of a statically defined window size used by normal scp. In addition, it provides a secure control channel over SSH and allows data to be transferred in cleartext to reduce overhead in unnecessary encryption. These characteristics allow bbftp to achieve bandwidths that are greater than normal scp.

bbFTP is recommended in place of *scp* for the data transfer of large files over long distances.

♦ *Where is it installed at NAS?*

Both the bbFTP server (*bbftpd*) and client (*bbftp*) are installed on all Columbia hosts, Lou[1-2], Pleiades front-end and bridge nodes (pfe[1-12], bridge[1-2]) and SUP.

For DMZFS[1,2], only the bbFTP server (*bbftpd*) is installed. Issue the *bbftp* command from Columbia, Pleiades, Lou or your localhost (if bbFTP client has been installed) to push files into DMZFS[1,2] or pull files out of DMZFS[1,2].

♦ *Do you need it installed on your localhost?*

If you want to initiate *bbftp* from your localhost, you have to download and install the client version of bbFTP on your localhost. If you want to initiate *bbftp* from a NAS HECC system and transfer files from/to your localhost, download and install the server version of bbFTP on your localhost.

♦ *When to use it?*

Consider using bbFTP when transferring large files (> 1 GB) within NAS or offsite. Be sure to use multiple streams to get better transfer rate.

- *Example:*

  bbftp is like a non-interactive ftp and the syntax can be complicated.

  ```
  your_localhost% bbftp -u nas_username -e 'setnbstream 8; get filename'
                  -E 'bbftpd -s -m 8' lou1.nas.nasa.gov
  ```

  ```
  For formatting issue, the above command was broken into
  two lines. In reality, it should be just one line.
  ```
- *Performance:*
    - ◊ bbFTP typically transfers data 10 - 20 times faster than normal scp.
    - ◊ Within NAS HECC Enclave, performance should be 100+ MB/sec.
    - ◊ Over WAN, the performance can be upto 50 MBytes/sec. File transfers between NAS and certain NASA sites may reach 100 Mbytes/sec.

      If you are not getting good performance, check with your network administrator to see if performance tuning is needed on your system.

The article bbFTP provides more instructions on installing and using bbFTP.

- **bbSCP**

bbSCP is written in Perl by Greg Matthews at NAS. It is a bbftp wrapper which provides an scp-like command-line interface. It assembles the proper command-line for bbftp and then executes bbftp to perform the transfers. bbSCP is designed and tested for bbftp version 3.2.0.

bbSCP only encrypts usernames and passwords, it does NOT encrypt the data being transferred.

- *Where is it installed at NAS?*

  bbSCP is installed on all Columbia hosts, Lou[1-2], Pleiades front-end and bridge nodes (pfe[1-12], bridge[1-2]) under /usr/local/bin.
- *Do you need it installed on your localhost?*

  If you want to initiate bbscp from your localhost, you need to:
    - ◊ download and install bbftp-client-3.2.0 on your localhost
    - ◊ download bbscp version 1.0.6 (also attached at the end of this article) and install it on your localhost
- *When to use it?*

Use the bbSCP script when you want the bbftp functionality and performance but with scp-like syntax. It can be used for transferring files within NAS HECC Enclave or between NAS and a remote site.

- *Example:*

  ```
  your_localhost% bbscp foo nas_username@lou1.nas.nasa.gov:
  ```
- *Performance:*

  Performance of bbSCP is the same as bbFTP.

The article bbscp provides more information (man page, performance turing, test and verification).

Outbound File Transfer Examples

DRAFT

This article is being reviewed for completeness and technical accuracy.

When the file transfer command (such as scp, bbftp or bbscp) is initiated on a NAS HECC host such as Columbia, Pleiades or Lou, the transfer does not need to go through SFE[1,2] or SUP. This is the fastest way to transfer files from/to your site if your localhost is configured to allow the transfer.

To simplify the instructions, the approaches will be described in terms of transfers to/from one of the Pleiades front-end node, pfe1, but they also apply to any of the other systems that are in the enclave (such as other Pleiades front-end or bridge nodes, Columbia or Lou). For each method described, two commands are provided. The first one is used when the user have identical username between his/her localhost and the NAS HECC systems. The second one is used when the usernames are different.

Logging into pfe1 and

- *Using scp for the outbound transfer:*

  To push files out of pfe1,

  ```
  pfe1% scp foo your_localhost:
  pfe1% scp foo local_username@your_localhost:
  ```

  To pull files into pfe1,

  ```
  pfe1% scp your_localhost:foo .
  pfe1% scp local_username@your_localhost:foo .
  ```

- *Using bbftp for outbound transfer:*

  If you find that using scp gives poor performance rates, we recommend using the application bbftp. This will require that the bbFTP server (bbftpd) is installed on your localhost.

  To push files out of pfe1,

  ```
  pfe1% bbftp -s -e 'setnbstream 8; put foo' your_localhost
  pfe1% bbftp -s -u local_username -e 'setnbstream 8; put foo' your_localhost
  ```

  To pull files into pfe1,

  ```
  pfe1% bbftp -s -e 'setnbstream 8; get foo' your_localhost
  ```

```
pfe1% bbftp -s -u local_username -e 'setnbstream 8; get foo' your_localhost
```

See bbftp for more instructions.

- *Using bbscp for outbound transfer:*

  bbSCP is a wrapper for bbFTP which provides scp-like syntax. Using this method for outbound transfer requires that the bbFTP server (bbftpd) is installed on your localhost.

  To push files out of pfe1,

  ```
  pfe1% bbscp foo your_localhost:
  pfe1% bbscp foo local_username@your_localhost:
  ```

  To pull files into pfe1,

  ```
  pfe1% bbscp your_localhost:foo .
  pfe1% bbscp local_username@your_localhost:foo .
  ```

See bbscp for more instructions.

Inbound File Transfer through SFEs Examples

DRAFT

This article is being reviewed for completeness and technical accuracy.

Inbound file transfers through SFEs require SecurID fob authentication, and the transfer can be done in two steps or one step depending on whether you have set up SSH passthrough.

To simplify the instructions, the approaches will be described in terms of transfers to/from one of the Pleiades front-end node, pfe1, but they also apply to any of the other systems that are in the enclave (such as other Pleiades front-end or bridge nodes, Columbia or Lou). For each method described, two commands are provided. The first one is used when (1) the user have identical username between his/her localhost and the NAS HECC systems, or (2) the usernames are different but the user has set up his/her local *~/.ssh/config* file to include the NAS username. To learn how to set this up, download the ~/.ssh/config template. The second one is used when the usernames are different and the user does not include the NAS username in his/her local *~/.ssh/config* file.

- Two-step file transfers

  If you have not set up SSH passthrough, this will be your only option for inbound file transfers. It requires you to use scp twice: once on your localhost to transfer files to/from one of the SFEs (for example, sfe1), and the second one on the SFE or the host inside the HECC Enclave to transfer files between SFEs and the HECC host such as pfe1.

  To push files out of your localhost,

  ```
  step 1:
  your_localhost% scp foo sfe1.nas.nasa.gov:
  your_localhost% scp foo nas_username@sfe1.nas.nasa.gov:

  step 2:
  sfe1% scp foo pfe1:
  or
  pfe1% scp sfe1:foo .
  ```

  To pull files into your localhost,

  ```
  step 1:
  sfe1% scp pfe1:foo .
  or
  pfe1% scp foo sfe1:

  step 2:
  ```

```
your_localhost% scp sfe1.nas.nasa.gov:foo .
your_localhost% scp nas_username@sfe1.nas.nasa.gov:foo .
```

- One-step file transfers

  If you have set up SSH passthrough correctly, you can use either scp, bbftp or bbscp to transfer files between your localhost and a NAS HECC host.

  - *Using scp,*

    To push files out of your localhost,

    ```
    your_localhost% scp foo pfe1.nas.nasa.gov:
    your_localhost% scp foo nas_username@pfe1.nas.nasa.gov:
    ```

    To pull files into your localhost,

    ```
    your_localhost% scp pfe1.nas.nasa.gov:foo .
    your_localhost% scp nas_username@pfe1.nas.nasa.gov:foo .
    ```

  - *Using bbftp,*

    This requires that you have a bbftp client installed on your localhost.

    To push files out of your localhost,

    ```
    your_localhost% bbftp -s -e 'setnbstream 8; put foo' pfe1.nas.nasa.gov
    your_localhost% bbftp -s -u nas_username
                    -e 'setnbstream 8; put foo' pfe1.nas.nasa.gov
    ```

    For formatting issue, the second command above was broken into two lines. In reality, it should be in one line.

    To pull files into your localhost,

    ```
    your_localhost% bbftp -s -e 'setnbstream 8; get foo' pfe1.nas.nasa.gov
    your_localhost% bbftp -s -u nas_username
                    -e 'setnbstream 8; get foo' pfe1.nas.nasa.gov
    ```

    For formatting issue, the second command above was broken into two lines. In reality, it should be in one line.

    See bbftp for more instructions.
  - *Using bbscp,*

    This requires that you have the bbftp client version 3.2.0 and the NAS bbscp script installed on your localhost.

To push files out of your localhost,

```
your_localhost% bbscp foo pfe1.nas.nasa.gov:
your_localhost% bbscp foo nas_username@pfe1.nas.nasa.gov:
```

To pull files into your localhost,

```
your_localhost% bbscp pfe1.nas.nasa.gov:foo .
your_localhost% bbscp nas_username@pfe1.nas.nasa.gov:foo .
```

See bbscp for more instructions.

Using the Secure Unattended Proxy (SUP)

The Secure Unattended Proxy (SUP) allows users to perform remote operations on specific hosts within the HEC enclave (currently the Columbia front-ends, Pleiades front-ends/bridge nodes, Lou[1-2], and Susan)

without

the use of SecurID at the time of the operation. Users must obtain special "SUP keys" using SecurID authentication, after which those keys can be used to perform operations from unattended jobs and/or scripts.

SUP keys are currently allowed to call *scp, sftp, bbftp, qstat, rsync,* and *test*. In the future, other operations may be available via the SUP. Each SUP key is valid for a period of **one week** from the time it is generated. Users may have multiple SUP keys at the same time, which will expire asynchronously.

SUP Usage Summary

The steps below demonstrate how to quickly get up and running with the SUP using an scp transfer to pfe1 as an example. Consult the link in each step for full details (or simply read this page to completion).

1. Download and install client (one time)

```
your_localhost% wget -O sup http://hecc.nas.nasa.gov/kb/file/9
your_localhost% chmod 700 sup
your_localhost% mv sup ~/bin
```

2. Authorize host for SUP operations (one time per host)

```
your_localhost% ssh pfe1
pfe1% touch ~/.meshrc
```

3. Authorize directories for writes (one or more times per host)

```
your_localhost% ssh pfe1
pfe1% echo /tmp >>~/.meshrc
```

4. Execute command (each time)

```
your_localhost% sup scp foobar pfe1:/tmp/c_foobar
```

5. Examine expected output (as needed)

6. Troubleshoot problems (as needed)

SUP Client

The SUP client performs all the steps needed to execute commands through the SUP as if the SUP itself did not exist. Commands that are allowed to pass through the SUP can be executed as if the remote host were directly connected by simply prepending the client command "sup". Besides executing remote commands, the client also includes an operating system-independent virtual file system that allows files across all SUP-connected resources to be accessed using standard filesystem commands.

- **Requirements**

  The client requires Perl version 5.6.1 or above to execute and has been tested successfully on Linux, OS X, and Windows under Cygwin and coLinux. Only SSH is required to use the SUP, however, so if these requirements cannot be met, it is possible to use the SUP without the client.

  Note for Windows users: even if the client is not used, scp and sftp require functionality only found in the OpenSSH versions of these commands, so Cygwin or coLinux will still be needed.
- **Installation**

  1. Download the client and save to a file called "sup"

  2. Make the client executable using "chmod 700 sup"

  3. Move the client to a location in your $PATH

- **SSH Configuration**

  If your local username differs from your NAS username, it is recommended that you add the following to your *~/.ssh/config* file, where "nas_username" should be replaced with your NAS username:

  ```
  Host sup.nas.nasa.gov sup-key.nas.nasa.gov
       User nas_username
  ```

  NOTE: If you are using a config file based on the NAS config template, you do not have to do this step.

  Alternatively, the client's -u option can be used as described in the next section. If your local username is the same as your NAS username, no additional configuration or command-line options are required.

- **SUP Command-line Options**

  ♦ -b

Disable user interaction for use within scripts. Note that the client will fail if any interaction is required - normally only needed when your SUP key has expired or is otherwise unavailable.

- -k

  By default, the client leaves any SSH agents started on your behalf running for future invocations after the client exits. This option forces spawned agents to be killed before exiting. Note that "-b" automatically implies "-k".

- -u user

  Specify NAS username. Note that this option is required if your local username differs from your NAS username and you have not modified your SSH configuration appropriately.

- -v

  Enable verbose output for debugging purposes.

SUP Authorizations

The basic set of operations that may be performed using the SUP is specified by the administrator. To protect accounts from malicious use of SUP keys, users must grant execute and write permissions to SUP operations on each target system.

1. **Execute Authorization**

   By default, even SUP operations permitted by site policy are not allowed to execute on a given host. To enable SUP operations to a given host (currently, the Columbia front-ends, Pleiades front-ends/bridge nodes, Lou[1-2], or Susan), the file *~/.meshrc* must exist on that host, which can be created by invoking the following:

   ```
   touch ~/.meshrc
   ```

   Note that the Pleiades front-ends/bridge nodes share their home filesystems, so this must only be done on one of these nodes. Similarly, the Columbia front-ends share their home filesystems and the *~/.meshrc* file only needs to be created on one of the Columbia front-end nodes. Other systems must be authorized separately. Once this file exists on a host, all operations permitted by site policy are allowed to execute on that host.

2. Write Authorization

By default, SUP operations are not allowed to write to the file system on a given host. To enable writes to a given directory on a given host, that directory must be added (on a separate line) to the *~/.meshrc* file on that host. For example, the following lines in *~/.meshrc* indicate that writes should be permitted to /nobackupp40 and /tmp.

```
/nobackupp40
/tmp
```

Each directory is the root of allowed writes, so this configuration would allow writes to all files and directories rooted at /nobackupp40 and /tmp (for example, /nobackupp40/some/dir, /tmp/some/file).

Note that the root directory cannot be authorized. Also note that dot files (i.e. *~/.**) in your home directory are never writable regardless of the contents of *~/.meshrc*.

Executing Commands Through SUP

Usage example of each command that may be executed through the SUP are given below. Note that SUP commands must be authorized for execution on each target host, and that transfers to a given host must be authorized for writes. Before a given operation is performed, the client may ask for certain information, including the existing or new passphrase for *~/.ssh/id_rsa*, the password + passcode for sup.nas.nasa.gov, and/or the password + passcode for sup-key.nas.nasa.gov.

File Transfer Commands

bbftp (man page)

```
your_localhost% sup bbftp -e "put foobar /tmp/c_foobar"
pfe1.nas.nasa.gov
```

Note that you must use the fully qualified domain name of the target host (in this case, pfe1.nas.nasa.gov) if you are not within the NAS domain.

bbscp (man page)

```
your_localhost% sup bbscp foobar pfe1.nas.nasa.gov:/tmp/c_foobar
```

Note that bbscp is just a client-side wrapper for bbftp, therefore, as with bbftp, you must use the fully qualified domain name of the target host (in this case, pfe1.nas.nasa.gov) if you are not within the NAS domain.

rsync (man page)

```
your_localhost% sup rsync foobar pfe1:/tmp/c_foobar
```

If you intend to transfer files to your home directory, note that even if your home directory has been authorized for writes, **rsync transfers to your home directory will fail unless the "-T" or "--temp-dir" option is specified**. This is because rsync uses temporary files starting with "." during transfers, which cannot be written in your home directory. You can avoid this problem by specifying an alternate temporary directory that is authorized for writes. For example, the following example uses /tmp as the temporary directory when files are transferred to the home directory. Make sure that the temporary directory specified has enough space for the files being transferred.

```
your_localhost% sup rsync -T /tmp foobar pfe1:
```

scp (man page)

```
your_localhost% sup scp foobar pfe1:/tmp/c_foobar
```

sftp (man page)

```
your_localhost% sup sftp pfe1
```

File Monitoring Command

test (man page)

```
your_localhost% sup ssh pfe1 test -f /tmp/c_foobar
```

Job Monitoring Command

qstat (man page available on Pleiades and Columbia)

```
your_localhost% sup ssh pfe1 qstat @pbspl1
```

SUP Expected Output

The following sequence shows the expected output for the command:

```
your_localhost% sup scp foobar pfe1:/tmp/c_foobar
```

for a user who has never used the SUP before.

The conditions under which each sub-sequence will be seen are indicated next to each header. Most of the items will only be seen once or during key generation. A second invocation will only show the command output portion.

1. **Host key verification** (seen once per client host)

```
No host key found for sup-key.nas.nasa.gov
...continue if fingerprint is
1b:9a:82:2b:b9:b0:7d:e5:08:50:1d:e8:14:76:a2:2e
The authenticity of host 'sup-key.nas.nasa.gov (129.99.242.7)'
can't be established.
RSA key fingerprint is
1b:9a:82:2b:b9:b0:7d:e5:08:50:1d:e8:14:76:a2:2e.
Are you sure you want to continue connecting (yes/no)? yes
No host key found for sup.nas.nasa.gov
...continue if fingerprint is
52:f3:61:9b:9c:73:79:4d:22:cb:f3:cd:9a:29:4e:fe
The authenticity of host 'sup.nas.nasa.gov (129.99.242.6)'
can't be established.
RSA key fingerprint is
52:f3:61:9b:9c:73:79:4d:22:cb:f3:cd:9a:29:4e:fe.
Are you sure you want to continue connecting (yes/no)? yes
```

2. **Identity creation** (seen during key generation if no identity available)

```
Cannot find identity /home/user/.ssh/id_rsa
...do you wish to generate it? (y/n) y
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
a3:cf:e5:50:12:6f:14:b1:21:59:19:a8:33:aa:77:40 user@host
```

3. **Identity addition to agent** (seen during key generation)

```
Adding identity /home/user/.ssh/id_rsa to agent
Enter passphrase for /home/user/.ssh/id_rsa:
Identity added: /home/user/.ssh/id_rsa
(/home/user/.ssh/id_rsa)
```

4. **Identity initialization** (seen once per identity)

```
Initializing identity on sup-key.nas.nasa.gov (provide login
information)
Password:
Enter PASSCODE:
```

```
Key a3:cf:e5:50:12:6f:14:b1:21:59:19:a8:33:aa:77:40 uploaded
successfully
```

5. **SUP key generation** (seen when no valid SUP keys available)

```
Generating key on sup.nas.nasa.gov (provide login information)
Password:
Enter PASSCODE:
```

6. **Client upgrade** (seen during key generation when new client available)

```
A newer version of the client is available (0.39 vs. 0.37)
...do you wish to replace the current version? (y/n) y
```

7. **Command output** (always seen)

```
foobar 100% 5 0.0KB/s 00:00
```

SUP Troubleshooting

The following error messages may be encountered during your SUP client usage. Note that the "-v" option can be given to the SUP client to output additional debugging information.

- "WARNING: Your password has expired"

  This message indicates that your current password has expired and must be changed. To change your password, you must log in to an LDAP host (for example, Lou) through the SFEs and change your LDAP password. This change will be automatically propagated to the SUP within a few minutes.

- "Permission denied (*~/.meshrc* not found)"

  This message indicates that you have not created a *.meshrc* file in your home directory on the target host. SUP commands must be authorized for execution on each target host.

- "Permission denied (unauthorized command)"

  This message indicates that you have attempted an operation that is not currently authorized by the SUP. Check that the command line is valid and that the attempted command is one of the authorized commands. Certain options to authorized commands may also be disallowed, but these should never be needed in standard usage scenarios.

- Permission denied during file access (various forms)

These messages indicate that you attempted to read or write a file for which such access is not allowed. The most common cause is forgetting to authorize directories for writes. Reads and writes of *~/.** are never permitted.

File Staging through DMZ File Servers

The NAS DMZ (Demilitarized Zone) file transfer servers, dmzfs1.nas.nasa.gov and dmzfs2.nas.nasa.gov, are designed to help facilitate file transfers into and out of the NAS enclave. All Lou users have an account on the DMZ file servers.

Design

- Each DMZ server is independent; they do not share filesystems or data.
- The DMZs do not support RSA SecurID authentication, so, the RSA key fob is not needed, and setting up SSH passthrough is not required. Instead, a password or public/private key pair should be used for authentication
- SCP and bbFTP are supported file transfer protocols.

Setup

To set up public key authentication for the DMZs, copy the public key, which you have likely already created on your local host, to the *authorized_keys* file of dmzfs1 and/or dmzfs2:

```
localhost% scp ~/.ssh/id_rsa.pub nas_username@dmzfs1.nas.nasa.gov:~/.ssh
localhost% ssh nas_username@dmzfs1.nas.nasa.gov
dmzfs1% cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

- Files should be pushed to or pulled from the DMZs.
- Unattended file transfers via the DMZs can be done with public key authentication. Files generated inside the NAS HECC Enclave can be pushed to the DMZ file servers under script control (but *not through PBS jobs*). Likewise, remote systems can automatically push files to the DMZ file servers. Then, scripts operating on Pleiades or Columbia can periodically check for file availability on the DMZ file servers, and when available, will pull the file into Pleiades or Columbia.

Restrictions

- The user environments are jailed; executable commands are minimal.

- Outbound connections are not allowed. File transfers via the DMZ file servers using commands such as *scp* or *bbftp* must be initiated from your local host or NAS systems (such as Pleiades, Columbia, Lou) *not* dmzfs1 or dmzfs2.

- Storage space is limited (users share 2.5TB), and files are meant to be stored for very short durations. Every hour, files older than 24 hours are automatically removed.

Examples

The following examples assume that: a) You want to push a file to dmzfs1 from your local host and pull the file from pfe1; b) You have not set up public key authentication for the DMZs. Thus, password authentication is used.

- Using *scp,* first copy the file to the DMZ:

  ```
  localhost% scp foo dmzfs1.nas.nasa.gov:
  Password: <-- type in your lou password
  foo 100% 764 0.8KB/s 00:00
  ```

  If your NAS username and local username are different:

  ```
  localhost% scp foo nas_username@dmzfs1.nas.nasa.gov:
  Password:   <-- type in your lou password
  foo                 100%  764     0.8KB/s   00:00
  ```

  then, you can pull the file from the DMZ:

  ```
  pfe1% scp dmzfs1:foo .
  Password:  <-- type in your lou password
  foo                 100%  764     0.8KB/s   00:00
  ```

- Using *bbftp,* first copy the file to the DMZ:

  ```
  localhost% bbftp -s -e 'put foo' dmzfs1.nas.nasa.gov
  Password: <-- type in your lou password
  foo 100% 764 0.8KB/s 00:00
  ```

  If your NAS username and local username are different:

  ```
  localhost% bbftp -s -u nas_username -e 'put foo' dmzfs1.nas.nasa.gov
  Password:   <-- type in your lou password
  put foo OK
  ```

  then, you can pull the file from the DMZ:

  ```
  pfe1% bbftp -s -e 'get foo' dmzfs1
  Password:   <-- type in your lou password
  get foo OK
  ```

  See the article on bbftp for more instructions.

bbftp

DRAFT

This article is being reviewed for completeness and technical accuracy.

When and Why to use bbFTP

If your data is being transferred to or from a NAS system over the wide area network, scp will almost always be the limiting factor, due to the static TCP windowing defined in the OpenSSH (versions older than 4.7) source code. The Bandwidth Delay Product (BDP) states that the bandwidth of the pipe multiplied by the latency gives the optimal window size for data transfer. With the window size statically defined for lower-speed networks, scp can never properly utilize the bandwidth available. bbFTP has dynamically adjustable window sizes (up to the maximum allowed by the system) and can also transmit multiple simultaneous streams of data. We have found that this application provides the best mechanism for making use of the bandwidth available between two sites.

Things to check:

- Are you using scp to transfer files?
- Are you transferring files to an offsite location? (outside NAS or NASA Ames)
- Is the average delay between sites larger than 30 ms?
- Is the data being transferred in large files (1 GB+)?

If the answer to all of these is 'Yes', then the bbFTP application will improve data transfer rates. Please follow the guide below to get started.

Downloading bbFTP

bbFTP has been tested to work on many operating systems: Linux, IRIX, Solaris, BSD and MacOSX. Other systems may also be supported.

If you intend to intiaite bbFTP from your localhost, you will need to install the bbFTP client on your localhost. If you intend to initiate bbFTP from a NAS host, you will need to install the bbFTP server on your localhost.

- bbFTP for Linux, IRIX, Solaris, and BSD

  For Linux, IRIX, Solaris, and BSD systems, the bbFTP application can be downloaded from its distribution site IN2P3 in France. For your convenience, the latest version is available here.

  Download latest client version - bbftp-client-3.2.0 (GZ compressed file - 232 KB)

Download latest server version - bbftp-server-3.2.0 (GZ compressed file - 220 KB)

- bbFTP for MacOSX

  Download latest client version with fixes for MacOSX (binary - 252KB)

  Download latest server version with fixes for MacOSX (binary - 192KB)

Installing bbFTP

If you download a source code distribution, follow the instruction below to build and install bbFTP. This guide covers the client setup only. Installing the server version is similar.

```
your_localhost% tar -zxvf bbftp*
your_localhost% cd bbftp*/bbftpc  (or bbftp*/bbftpd  for the server version)
your_localhost% ./configure
your_localhost% make
your_localhost% make install (optional, requires root privileges to install)
```

By default, the application will install in /usr/local/bin. If you do not have admin privileges, you may skip the last step and copy the bbFTP binary to your home directory, or run it from the current location.

Using bbFTP

To write the version of bbftp and default values to standard output:

```
bbftp -v
```

For example:

```
pfe1% bbftp -v
bbftp version 3.2.0
Compiled with  :   default port 5021
                   compression with Zlib-1.2.3
                   encryption with OpenSSL 0.9.8a 11 Oct 2005
                   default ssh command = ssh -q
                   default ssh remote command = bbftpd -s
                   default number of tries = 5
                   default sendwinsize = 256 Kbytes
                   default recvwinsize = 256 Kbytes
                   default number of stream = 1
```

To request the execution of commands contained in the control file ControlFile or the ControlCommands using RemoteUsername on RemoteHost:

```
bbftp [Options] [-u RemoteUsername] -i ControlFile [RemoteHost]
bbftp [Options] [-u RemoteUsername] -e ControlCommands [RemoteHost]
```

Notice that -i or -e option are mandatory. The examples given in this article all use -e *ControlCommands*.

Available options are:

```
[-b (background)]
[-c (gzip compress)]
[-D[min:max] (Domain of Ephemeral Ports) ]
[-f errorfile]
[-E server command for ssh]
[-I ssh identity file]
[-L ssh command]
[-s (use ssh)]
[-S (use ssh in batch mode)]
[-m (special output for statistics)]
[-n (simulation mode: no data written)]
[-o outputfile]
[-p number of // streams]
[-q (for QBSS on)]
[-r number of tries ]
[-R .bbftprc filename]
[-t (timestamp)]
[-V (verbose)] will print out the transfer rate
[-w controlport]
[-W (print warning to stderr) ]
```

For more information about each option, see **man bbftp**. Those used in the examples will be briefly described.

Single stream vs multiple streams

- Single stream:

Using single stream is the easiest, but may not provide optimal performance.

In the examples below, bbFTP is run from the current working directory. If it was installed in a system path location, the "./" may be omitted.

The -s option says to use ssh to remotely start a bbftpd daemon. It usually starts the binary "bbftpd -s", but this can be changed througth the -E option.

The first command is to pull a file from a remotehost using *get* and the second command is to push a file to the remote host using *put*.

```
./bbftp -s -u remote_username -e 'get filename' remotehost
./bbftp -s -u remote_username -e 'put filename' remotehost
```

- Multiple streams:

For transfers between two NAS hosts, such as Pleiades and Lou, no more than 2 streams should be used.

For transfers between your site and NAS, more streams will probably help. In several tests, using 8 streams gave the best performance.

If there is little increase in the transfer rate from single stream to multiple streams, a lower number may be used. The value must be changed in both the control command (-e) and the server command (-E) so that the server listens for the same number of streams as the client requests.

In the examples below, -s is not used. Instead, -E 'bbftpd -s' is used to use ssh to remotely start a bbftpd daemon.

```
./bbftp -u remote_username -e 'setnbstream 8; get filename'
        -E 'bbftpd -s -m 8' remotehost
./bbftp -u remote_username -e 'setnbstream 8; put filename'
        -E 'bbftpd -s -m 8' remotehost
```

For formatting issue, each command above was broken into two lines. In reality, it should be just one line.

- *File related commands*

  You may need to use the command 'cd' to change directory on the remotehost or 'lcd' to change directory on the host where bbftp is issued in order to 'get' or 'put' files from/to the directory you intend to use. For the rules, please see the man page of bbftp. Here are some examples:

  ```
  bbftp -s -u remote_username
        -e 'cd /u/username/abc; get filename' remotehost
  bbftp -s -u remote_username
        -e 'cd /u/username/abc; lcd def; put filename' remotehost
  ```

  For formatting issue, each command above was broken into two lines. In reality, it should be just one line.
- *Initiating bbftp from a host outside of NAS domain*

  If you want to initiate bbftp from a host that is not within the NAS domain to transfer files to/from a NAS host (not including dmzfs1 and dmzfs2), you must do the following:

  Set up SSH passthrough.

  In the *.ssh/config* file on your localhost, be sure to include entries with the **fully-qualified domain name**. For example:

  ```
  Host pfe1.nas.nasa.gov
  ```

```
    ProxyCommand ssh sfe1.nas.nasa.gov /usr/local/bin/ssh-proxy pfe1.nas.nasa.gov
```

In the bbftp command line, use the **fully-qualified domain name** (ex: pfe1.nas.nasa.gov) of the NAS host. For example,

```
your_localhost% ./bbftp -s -u nas_username -e 'get filename'
pfe1.nas.nasa.gov
```

These two steps are needed due to the fact that bbftp uses 'gethostbyname' function to check a hostname for connection and then it uses ssh to connect to that hostname. Thus a fully-qualified domain name in the *./ssh/config* file is required. If the fully-qualified domain name cannot be found in *./ssh/config*, one will get the error:

```
BBFTP-ERROR-00061 : Error waiting MSG_LOGGED_STDIN message
```

For Pleiades, one has to use pfe[1-12].nas.nasa.gov or bridge[1-2].nas.nasa.gov. The front-end load balancer, **pfe.nas.nasa.gov**, does not work with bbftp. For example:

```
your_localhost% bbftp -s -u nas_username -e 'get filename' pfe.nas.nasa.gov
BBFTP-ERROR-00017 : Hostname no found (pfe.nas.nasa.gov)
```

On the other hand, for ssh or scp, one can use either the fully-qualified domain name above or the abbreviated name below:

```
Host pfe1
ProxyCommand ssh sfe1.nas.nasa.gov /usr/local/bin/ssh-proxy pfe1.nas.nasa.gov
```

- Specifying port range

Performance Tuning

To find the transfer rate, turn on the -V option.

Performance of bbFTP is affected by the number of streams and the TCP window sizes.

The TCP window size determines the amount of outstanding data a transmitting end-host can send on a particular connection before it gets acknowledgment back from the receiving end-host. For optimal performance, the window size should be set to the value of the Bandwidth Delay Product (i.e., the product of the bandwidth of the pipe and the latency).

bbFTP is compiled with a default send and receive TCP window size as can be seen with the -v option and can dynamically adjust the window size (up to the maximum allowed by the system) for better performance. However, a user can also choose a non-default send/recv window size (in KB). For example:

```
bbftp -e 'setrecvwinsize 1024; setsendwinsize 1024; put filename'
      -E 'bbftpd -s' remotehost
```

```
For formatting issue, the command above was broken into two lines.
In reality, it should be just one line.
```

For high-speed links where bbFTP is still not performing as well as expected, it may be due to a system windowing limitation. Most operating systems have the maximum window size set to a small value, such as 64 KB. As practice, NAS systems are set to a minimum of 512 KB.

If you are not gettting good performance, ask your local system administrator if performance tuning is necessary for your localhost.

The bbscp Script

DRAFT

This article is being reviewed for completeness and technical accuracy.

Introduction

The bbscp script is written in Perl by Greg Matthews at NAS. It is a bbftp wrapper which provides an scp-like command line interface; bbscp only encrypts usernames and passwords, it does *not* encrypt the data being transferred.

Downloading bbscp

If you plan to initiate bbscp on your localhost, you have to download bbscp version 1.0.6 (also attached at the end of this article) and download/install bbFTP client version 3.2.0 on your localhost.

The bbscp script has been installed on Pleiades (version 1.0.4), Columbia (version 1.0.4), and Lou (version 1.0.6).

Using bbSCP

Note that bbscp is just a client-side wrapper for bbftp, so, as with bbftp, you must use the *fully-qualified domain name* of the target host (for example, pfe1.nas.nasa.gov) if you are not within the NAS domain.

The bbSCP version 1.0.6 man page provides details on how to use it.

```
BBSCP(1)           User Contributed Perl Documentation              BBSCP(1)

NAME
       bbscp - bbftp wrapper, provides an scp-like commandline interface

SYNOPSIS
       bbscp [OPTIONS] [[user@]host1:]file_or_dir1 [...] [[user@]host2:]dir2

DESCRIPTION
       bbscp does unencrypted copies of files either from the localhost to a
       directory on a remote host, or from a remote host to a directory on
       the localhost (see the -N option for the only exception to this). It
       assembles the proper commandline for bbftp (designed and tested for
       bbftp version 3.2.0, see RESTRICTIONS) and then executes bbftp to
       perform the transfer(s).
```

```
        The "-s", "-p 2", and "-r 1" options for bbftp are set by default,
        along with the following options:

          setoption keepaccess
          setoption keepmode
          setoption nocreatedir

        The options -p and -r can be overridden on the commandline.

        Note the following limitations and capabilities in different transfer
        scenarios:

        copying from localhost to remote host
            - regular files
                bbftp will overwrite a pre-existing file of the same name on
                the remote host without asking for confirmation.

            - directories
                This script recursively transfers entire directories (only for
                local-to-remote transfers!).

            - symbolic links (see RESTRICTIONS)
                Symlinks on the localhost are treated just like the thing they
                point to, and are ignored if they point to something that
                doesn't exist.

        copying from remote host to localhost
            - regular files
                bbftp will overwrite a pre-existing file of the same name on
                the localhost without asking for confirmation.

            - directories
                There is no way at this time to transfer entire directories
                from a remote host to the localhost.

            - symbolic links (see RESTRICTIONS)
                Symlinks on the remote host are treated just like the thing
                they point to (which means they are ignored if they point to
                a directory or to something that doesn't exist).

OUTPUT
        The default output mode of the script displays "OK" or "FAILURE" for
        each of the transfer operations that bbftp performs. This display
        occurs after bbftp has finished running, so it may be delayed for
        some time depending on the duration of the transfer(s).

        The script switches to more verbose output if the user provides 1 or
        more of the verbose output commandline options (-l, -t, -V, and -W).

OPTIONS
          -B    name/location of bbftp executable. default is "bbftp"

          -d    dry-run. script performs its duty but does not actually
                execute bbftp. the bbftp commandline is printed, along
                with the contents of the bbftp control-file
          -h    minimal help text

          -k    keep bbftp command file that this script creates

          -l    long-winded (extra verbose) output from bbftp. uses
                undocumented bbftp option (-d)

          -N    transfer a single file and rename it at the destination.
                both local-to-remote and remote-to-local transfer is
                supported. see RESTRICTIONS

          -v    version of this script

          -X    set the size of the TCP send window (in kilobytes). default
                is the bbftp default size

          -Y    set the size of the TCP receive window (in kilobytes). default
                is the bbftp default size

          -z    suppress the security disclaimer

        bbftp options that can be specified on the commandline of this script:

          -D[min_port:max_port]   (e.g. "-D", "-D40000:40100")

          -E <Server command to run>

          -L <SSH command>

          -p <number of parallel streams>

          -R <bbftprc file>

          -r <number of tries>

          -t

          -V
          -W

RESTRICTIONS
        Version of bbftp
          It's very important to use bbftp version 3.2.0 with bbscp --
          there's at least 1 known issue with using bbftp 3.1.0.

        Possible shell issues
          bash and tcsh interpret commandline text in different ways, so you
          may need to use quotes or other delimiters to use bbscp. In
          particular, bash and tcsh are known to handle wildcards differently.

        Wildcards
          If the -N option is not in use, wildcards can be used in remote host
          file specifications, but only for the names of files, not for
          directories. So, for example, "user@host:/tmp/file*" is acceptable,
          but "user@host:/tm*/file*" is not.
```

```
       Symbolic links
         Symlinks are not bbftp's strong suit -- if you wish to transfer a
         collection of files that includes symlinks it is highly recommended
         that you first make a tar-file and then transfer the tar-file.

       Use of -N option
         Wildcards are not supported in remote host file specifications w/ -N.

         If the destination is a symlink it will be overwritten, regardless of
         what that symlink points to.

EXAMPLES
    Note: these examples have been tested with bash, changes may be needed for
    them to work in tcsh (see RESTRICTIONS).

    local file to remote directory (username must be the same on both machines)
         bbscp /u/username/data/file1 machine:target_dir

    local file to remote file w/ different name
         bbscp -N /u/username/data/file1 machine:file89

    multiple local files to remote directory
         bbscp /u/username1/data/*file username2@machine:/tmp

    local directory to remote home directory
         bbscp /u/username1/data username2@machine:

    remote file to local directory
         bbscp username1@machine:data/file5 /u/username2/source_dir

    remote file to local file w/ different name
         bbscp -N username1@machine:data/file5 /u/username2/source_dir/file93

    multiple remote files to local directory
         bbscp -V username1@machine:/u/username1/data/file* /tmp

    multiple remote files to local directory
         bbscp -V username1@machine:file1.txt username1@machine:stuff.dat /tmp

AUTHOR
      Greg Matthews gregory.matthews@nasa.gov

perl v5.8.8       2010-12-10    BBSCP(1)
```

Performance Tuning

To find the transfer rate, turn on -V option.

Like bbftp, the number of streams and TCP send/recv window sizes affect performance. Users can set the number of streams through the -p option. Starting with bbscp version 1.0.6, *the default is 2 streams***.** To set the window sizes in KB, use the *-X* option for send window and *-Y* for receive window. The efault is the bbftp default send/recv window size.

Test and Verification

bbscp man page

DRAFT

This article is being reviewed for completeness and technical accuracy.

The man page for bbscp as seen on Lou.

Using bbscp for Test and Verification

DRAFT

This article is being reviewed for completeness and technical accuracy.

The following examples provide test and verification data and sample commands for using bbscp between two hosts (crow & cfe3.nas.nasa.gov or dmzfs1.nas.nasa.gov).

Straight file transfer

This example demonstrates the tranfer of a file named *100mb*.

```
crow% bbscp -V 100mb user@cfe3.nas.nasa.gov:/nobackup1/user/

/home/user/bin/bbscp: will run commandline:
   bbftp -s -r 1 -V -p 8 -u user -i /tmp/bbscp.lKCrSUg cfe3.nas.nasa.gov

/home/user/bin/bbscp: begin output of bbftp:

-----------------------------------------------------------------
WARNING! This is a US Government computer. This system is for
.....
-----------------------------------------------------------------
Authenticated with partial success.

Plugin authentication

Enter PASSCODE:

>> COMMAND : setoption keepaccess
<< OK
>> COMMAND : setoption keepmode
<<  OK
>> COMMAND : setoption nocreatedir
<<  OK
>> COMMAND : put 100mb /nobackup1/user/100mb
<<  OK
104857600 bytes send in 5.43 secs (1.89e+04 Kbytes/sec or 147 Mbits/s)

/home/user/bin/bbscp: end output of bbftp
```

Renaming file at destination

Transfer a single file (named *100mb*) and rename it (to *crow-100mb*) at the destination; both local-to-remote and remote-to-local transfer is supported.

```
crow% bbscp -V -N 100mb user@cfe3.nas.nasa.gov:/nobackup1/user/crow-100mb

/home/user/bin/bbscp: will run commandline:
                      bbftp -s -r 1 -V -p 8 -u user -i

/tmp/bbscp.5eUBcTX cfe3.nas.nasa.gov

/home/user/bin/bbscp: begin output of bbftp:

----------------------------------------------------------------

WARNING! This is a US Government computer. This system is for
.....
----------------------------------------------------------------

Authenticated with partial success.

Plugin authentication

Enter PASSCODE:

>> COMMAND : setoption keepaccess
<< OK
>> COMMAND : setoption keepmode
<< OK
>> COMMAND : setoption nocreatedir
<< OK
>> COMMAND : put 100mb /nobackup1/user/crow-100mb
<< OK
104857600 bytes send in 5.3 secs (1.93e+04 Kbytes/sec or 151 Mbits/s)

/home/user/bin/bbscp: end output of bbftp
```

Adjusting the TCP window size

This example demonstrates the use of -X and -Y options to set the TCP window size (available in bbscp Version 1.0.2 and above).

```
crow% ./bbscp -V -N -X 2000 -Y 2000 1gig.dat user@dmzfs1.nas.nasa.gov:/home/user/garbag

bbscp: will run commandline:
       bbftp -s -r 1 -V -p 8 -u kfreeman
             -i /tmp/bbscp.SNxL5RT dmzfs1.nas.nasa.gov

bbscp: begin output of bbftp:

user@dmzfs1.nas.nasa.gov's password:

>> COMMAND : setoption keepaccess
<< OK
>> COMMAND : setoption keepmode
```

```
<< OK
>> COMMAND : setoption nocreatedir
<< OK
>> COMMAND : setsendwinsize 2000
<< OK
>> COMMAND : setrecvwinsize 2000
<< OK
>> COMMAND : put 1gig.dat /home/kfreeman/garbage.dat
<< OK

1109393408 bytes send in 34.6 secs (3.13e+04 Kbytes/sec or 244 Mbits/s)

bbscp: end output of bbftp
```

Dry run/debugging

This example demonstrates the use of the -d option for dry run. In this case, the bbscp script performs its duty but does not actually execute bbFTP. The bbFTP command line is printed, along with the contents of the bbFTP control-file.

```
cfe3.user% bbscp -d -V -N one-gig user@crow.eos.nasa.gov:/home/user/data/cfe3-one-gig
/usr/local/bin/bbscp: would have run commandline:
                      bbftp -s -r 1 -V -p 8 -u user
                      -i /tmp/bbscp.4PZYIuL crow.eos.nasa.gov

/usr/local/bin/bbscp: bbftp control-file (/tmp/bbscp.4PZYIuL) looks like:

setoption keepaccess
setoption keepmode
setoption nocreatedir
put one-gig /home/user/data/cfe3-one-gig
```

Using the SUP Virtual File System

DRAFT

This article is being reviewed for completeness and technical accuracy.

Introduction

The SUP client includes a virtual file system (VFS) capability that allows files across all SUP connected resources to be accessed using standard file system commands. For example, the command:

```
ls /sup/pfe1/tmp
```

would list the files in /tmp on *pfe1*. The command:

```
cp foobar /sup/pfe1/tmp
```

would copy the file "foobar" from the current directory on the local host to /tmp on *pfe1*.

The set of supported commands includes cat, cd, chgrp, chmod, chown, cmp, cp, df, diff, du, file, grep, head, less, ln, ls, mkdir, more, mv, pwd, rm, rmdir, tail, tee, test, touch, and wc. Note that this functionality is not a true file system since only these commands are supported and only when used from within a shell. Unlike more general approaches such as FUSE, however, the SUP capability is completely portable and can be enabled with no additional privileges or software.

Commands through the VFS functionality can act on any combination of local and remote files, where remote files are prefixed with "/sup/hostname". For example, the command:

```
cat /sup/pfe1/tmp/rfile ~/lfile
```

would print the file "rfile" in /tmp on *pfe1* as well as the file "lfile" in the user's home directory on the local host to the terminal. Any number of hosts can be included in any command. For example, the command:

```
diff /sup/pfe1/tmp/cfe_file /sup/pfe/tmp/pfe_file
```

would show the differences between the file "cfe_file" in /tmp on *pfe1* and the file "pfe_file" in /tmp on pfe. The client determines if any remote access is needed based on the path(s) given. If not, it will execute the command locally as given as rapidly as possible. Fully local commands also support all options with the exception of options of the form "-f value" (i.e. single-dash options that take values).

VFS Activation

- Requirements

  Currently, SUP VFS functionality is **only supported for bash**, but csh support is planned for the future. This functionality **requires Perl version 5.8.5** (note that this is more recent than version 5.6.1 required by the basic client functionality). It also **requires the standard Unix utilities cat, column, false, sort, and true** and has been tested successfully on Linux, OS X, and Windows under Cygwin and coLinux. Note that users of Windows under Cygwin may need to install the coreutils and util-linux packages to obtain these utilities.
- Activation/Deactivation

  1. Install the SUP client if you have not already done so
  2. Activate VFS functionality in a bash shell

     ```
     eval `sup -s bash`
     ```

     This will load aliases and functions used to intercept specific commands and replace them with commands through the SUP client that perform the actions requested.

  3. Deactivate VFS functionality in a bash shell whenever desired

     ```
     eval `sup -r bash`
     ```
- Command-line Options

  The behavior of the virtual file system can be modified using various options at the time it is activated.

  ♦ -m /newroot

    Change the root of the virtual file system from its "/sup" default to "/newroot".

  ♦ -ocmd=opts

    Specify default options for a given command since the VFS functionality overrides any existing aliases for its supported set of commands.

  ♦ -t transport

    Change the file transport from its "sftp" default to "transport". Currently, the only additional transport available is "bbftp". Note that using bbftp as the transport may slow down certain operations on small files as bbftp has higher startup overhead.

  ♦ -u user

Specify NAS user name. Note that this option **is required if your local user name differs from your NAS user name**.

For example, the following invocation activates the client virtual file system using bbftp as the transport mechanism, "nasuser" as the user and adds colorization of local file listings using the Linux ls "--color=always" option.

```
eval `sup -s bash -t bbftp -u nasuser -ols=--color=always`
```

VFS Caveats

The VFS functionality is still somewhat experimental. In general, it works for the most common usage scenarios with some caveats. In particular:

- "Whole file" commands (i.e. commands that must process the entire file), including cat, cmp, diff, grep, wc (and currently more/less due to implementation) retrieve files first before processing for efficiency. Thus, these commands should not be executed on very large files.
- There is a conflict between commands that take piped input and the custom globbing of the client, thus these commands have portions of globbing support disabled. These commands are grep, head, less, more, tail, tee, and wc. In these cases, globbing will work for absolute prefixes, but not relative. For example, "grep foo /sup/pfe1/tmp/*" will work, but "cd /sup/pfe1/tmp; grep foo *" will not.

- Redirection to/from remote files doesn't work. The same effect can be achieved using cat and tee (e.g. "grep localhost a" would become "cat /sup/pfe1/etc/hosts |grep localhost |tee a >/dev/null"). Redirection still works normally for local files.
- The directories "/sup" and "/sup/hostname" show up in neither completions nor ls, so you must know they exist.
- The first time a command is run involving a particular host, a SFTP connection is created to that host. When running "ps", it may appear as if a zombie client process is running.
- Commands may hang the first time after switching networks (e.g. with a laptop). If this happens, hit Control-c and it will work the next time.

VFS Commands

Currently supported commands and their currently supported options are below. Unsupported options will simply be ignored except where noted. All commands are still subject to SUP authorizations, thus something that cannot be executed or written normally through the SUP cannot be executed or written through this functionality either.

- **cat (no options)**
- **cd (no options)**

Note that when changing to remote directories, cd only changes $PWD so to make changes visible, the working directory (i.e. \w in bash) must be in your prompt. For example, the following prompt:

```
export PS1="\h:\w> "
```

would display the current host name followed by the current working directory.

- **chgrp (no options)**

  Groups may be specified either by number or by name. Names will be resolved on the remote host.

- **chmod (no options)**

  Modes must be specified numerically (e.g. 0700). Symbolic modes, such as a+rX, are not currently supported.

- **chown (no options)**

  Users and groups may be specified either by number or by name. Names will be resolved on the remote host.

- **cmp (all options)**
- **cp [-r]**

  Note that copies between two remote hosts transfer files to the local host first since the SUP does not allow third party transfers. Thus, very large file transfers between remote systems should be achieved using an alternate approach.

- **df [-i]**

  Note that 1024-byte blocks are used.

- **diff (all options)**
- **du [-a] [-b] [-s]**

  Note that 1024-byte blocks are used.

- **file (all options)**

- **grep (all options)**

- **head [-number]**

  Note that head does not support the form "-n number", thus, for example, to display the first 5 lines of a file, use "-5" and not "-n 5".

- **less (all options)**
- **ln [-s]**

  Note that hard links are not supported. Links from remote files to local files (e.g. ln -s /sup/pfe1/foo /foo) will be dereferenced during certain operations (e.g. cat /foo will cat /sup/pfe1/foo).

- **ls [-1] [-d] [-l]**

  For efficiency purposes, ls behaves slightly differently for remote commands than for local. In particular "ls -l" will not show links by default and will show what is actually linked instead of the link itself. Link details can be obtained using the "-d" option (e.g. ls -ld *).

  Also for efficiency, ls processes remote files before local files, so output ordering may be changed when remote and local files are interleaved on the ls command line. For example, "ls /foo/sup/pfe1/bar" would show /sup/pfe1 first, then /foo, then /bar.
- **mkdir (no options)**
- **more (all options)**
- **mv (no options)**
- **pwd (no options)**
- **rm [-r]**
- **rmdir (no options)**
- **tail [-number]**

  Note that tail does not support the form "-n number", thus, for example, to display the last 5 lines of a file, use "-5" and not "-n 5".

- **tee [-a]**
- **test [-b] [-c] [-d] [-e] [-f] [-g] [-h] [-k] [-L] [-p] [-r] [-s] [-S] [-u] [-w]**

  Note that compound and string tests are not supported. Compound and string tests can be achieved using multiple test commands separated by shell compound operators. For example,

  ```
  test -f /sup/pfe1/foo -a "abc" != "123"
  ```

  would become

```
test -f /sup/pfe1/foo && test "abc" != "123"
```

Alternatively, the "actual" test command can be executed through the SUP:

```
sup ssh pfe1 test -f /foo -a "abc" != "123"
```

- **touch (no options)**
- **wc (all options)**

Using the SUP without the SUP Client

DRAFT

This article is being reviewed for completeness and technical accuracy.

Introduction

The SUP client is the recommended approach to using the SUP. The client requires Perl, however, thus may not be suitable for all purposes. The only software actually required to use the SUP is SSH. This page details the manual steps required to use the SUP with only SSH. Users should still review the client instructions for a full overview of the SUP.

SUP Manual Usage Summary

The steps below demonstrate how to get up and running with the SUP without the client using a bbftp transfer to cfe1 as an example. Consult the link in each step for full details (or simply read this page to completion).

1. Initialize a long-term key on sup-key.nas.nasa.gov (one time)

   ```
   ssh -x -oPubkeyAuthentication=no sup-key.nas.nasa.gov \
       mesh-keygen --init <~/.ssh/authorized_keys
   ```
2. Generate a SUP key (one time per week)

   ```
   eval `ssh-agent`
   ssh-add ~/.ssh/id_rsa
   ssh -A -oPubkeyAuthentication=no sup.nas.nasa.gov \
       mesh-keygen |tee ~/.ssh/supkey`
   ssh-agent -k
   ```
3. Authorize host for SUP operations (one time per host)

   ```
   ssh cfe1
   touch ~/.meshrc
   ```
4. Authorize directories for writes (one or more times per host)

   ```
   ssh cfe1
   echo /tmp >>~/.meshrc
   ```
5. Prepare the SUP key for use (one time per session)

   ```
   eval `ssh-agent`
   ssh-add -t 1w ~/.ssh/supkey
   ```
6. Execute command (each time)

   ```
   bbftp -L "ssh -Aqx -oBatchMode=yes sup.nas.nasa.gov ssh -q" \
       -e "put /foo/bar /tmp/c_foobar" cfe1.nas.nasa.gov
   ```

7. Troubleshoot problems (as needed)

SUP Key Generation

1. **On the very first use only**, invoke the "mesh-keygen" command with the "--init" option on sup-key.nas.nasa.gov to upload an SSH authorized_keys file (used *only* during key generation and revocation). An authorized_keys file contains one or more SSH public keys that allow the corresponding SSH private keys to be used for authentication to a system. The uploaded authorized_keys file can be an existing file (such as your ~/.ssh/authorized_keys file from any host) or one created specifically for this purpose using a new SSH key pair generated with ssh-keygen. The public keys in this file must be in OpenSSH format (i.e. *not* the format of the commercial SSH version used on the Secure Front-Ends [SFEs]) and must not contain any forced commands (i.e. "command="). For example, to upload an existing authorized_keys file, the following can be invoked:

   ```
   ssh -x -oPubkeyAuthentication=no sup-key.nas.nasa.gov \
       mesh-keygen --init <~/.ssh/authorized_keys
   ```

   You will be prompted to authenticate using both a password (originally your Lou password) and securID passcode (PIN + tokencode).

   Users who have never connected to sup-key.nas.nasa.gov before may need to add a "-oStrictHostKeyChecking=ask" option to the scp command line. (RSA key fingerprint of sup-key.nas.nasa.gov is 1b:9a:82:2b:b9:b0:7d:e5:08:50:1d:e8:14:76:a2:2e)

   Note that this is on sup-key *only* and that you must use the "-oPubkeyAuthentication=no" option as shown. Users outside NAS may need to add an appropriate SSH option to set their login name, such as "-l username".
2. Start an SSH agent (or use one currently running):

   ```
   eval `ssh-agent -s` (if your shell is sh/bash)
   ```

   or

   ```
   eval `ssh-agent -c` (if your shell is csh/tcsh)
   ```

3. Add a private key corresponding to one of the public keys in the authorized_keys file of step 1 to the agent (this is unnecessary if an agent is already running with the key loaded). For example:

   ```
   ssh-add ~/.ssh/id_rsa
   ```

4. Invoke the "mesh-keygen" command on sup.nas.nasa.gov. You will be prompted to authenticate using both password (originally your Lou password) and securID passcode (PIN + tokencode). After successful authentication, the mesh-keygen

command prints a SUP key to your terminal, which should be saved to a file in a directory that is readable only by you. This key can be saved to a file by cut-and-paste, redirecting standard output, or using the "tee" command. For example, to generate a key and redirect it into a file starting with ~/.ssh/supkey and labeled with the current time, the following can be invoked:

```
ssh -A -oPubkeyAuthentication=no sup.nas.nasa.gov \
    mesh-keygen |tee ~/.ssh/supkey.`date +%Y%m%d.%H%M`
```

Users who have never connected to sup.nas.nasa.gov before may need to add a "-oStrictHostKeyChecking=ask" option to the SSH command line. (RSA key fingerprint of sup.nas.nasa.gov is 52:f3:61:9b:9c:73:79:4d:22:cb:f3:cd:9a:29:4e:fe)

Note that you must use the "-oPubkeyAuthentication=no" option as shown. Users outside NAS may need to add an appropriate SSH option to set their login name, such as "-l username".

5. Protect your keys. In order to perform unattended operations, SUP keys cannot be encrypted, thus should always be protected with appropriate file system permissions (i.e. 400 or 600). Check the permissions of your key immediately after generation and modify if necessary. You are responsible for the privacy of your keys.

SUP Key Management

Each invocation of mesh-keygen creates a new SUP key that is valid for one week from the time of generation. Users may have multiple keys at once that all expire at different times. To facilitate the management of multiple SUP keys, the "mesh-keytime" and "mesh-keykill" commands are available.

Mesh-keytime

To determine the expiration time of a SUP key stored in a file "/key/file", the following can be invoked:

```
ssh -xi /key/file -oIdentitiesOnly=yes -oBatchMode=yes \
    sup.nas.nasa.gov mesh-keytime
```

The key fingerprint and expiration time will be printed to your terminal.

Mesh-keykill

To invalidate a specific SUP key stored in a file "/key/file" before its expiration time has passed, you must have an SSH agent running with the same key you use to generate SUP keys as described in steps 2 and 3 of the SUP Key Generation section. After which, the following can be invoked:

```
ssh -Axi /key/file -oIdentitiesOnly=yes -oBatchMode=yes \
            sup.nas.nasa.gov mesh-keykill
```

To invalidate all currently valid SUP keys, the following can be invoked:

```
ssh -Ax -oPubkeyAuthentication=no sup.nas.nasa.gov mesh-keykill --all
```

In this case, you will be prompted to authenticate using both password and securID passcode.

SUP Key Preparation

Currently, the only operations allowed with a SUP key are scp, sftp, bbftp, qstat, rsync, and test. For all operations, an SSH agent must be started with the SUP key loaded, which can be scripted as needed, because the key is unencrypted.

1. Start an SSH agent:

   ```
   eval `ssh-agent -s` (if your shell is sh/bash)
   ```

   or

   ```
   eval `ssh-agent -c` (if your shell is csh/tcsh)
   ```
2. Add a SUP key to the agent (this is the *only* key required to perform unattended SUP operations):

   ```
   ssh-add /key/file
   ```

   Since SUP keys have a lifetime of one week, the "-t" option may be used to automatically remove the key from the agent after a week has elapsed:

   ```
   ssh-add -t 1w /key/file
   ```

   The will prevent a buildup of keys in the agent, which can cause login failure as described in the SUP Troubleshooting section. Keys may be explicitly removed from the agent using the following:

   ```
   ssh-keygen -y -f /key/file >/key/file.pub
   ssh-add -d /key/file
   ```
3. Make sure agent forwarding and batch mode are enabled in your SSH client. The examples below include the appropriate options to enable agent forwarding ("-A") and batch mode ("-oBatchMode=yes").

SUP Commands

Examples of the use of each command that may be executed through the SUP are given below. Note that SUP commands must be authorized for execution on each target host and transfers to a given host must be authorized for writes.

- **bbftp** (man page)

```
bbftp -L "ssh -Aqx -oBatchMode=yes sup.nas.nasa.gov ssh -q" \
    -e "put /foo/bar /tmp/c_foobar" cfe1.nas.nasa.gov
```

Note that **you must use the fully-qualified domain name of the target host** (in this case, cfe1.nas.nasa.gov) if you are not within the NAS domain.

- **bbscp** (man page)

```
bbscp -L "ssh -Aqx -oBatchMode=yes sup.nas.nasa.gov ssh -q" \
    foobar cfe1.nas.nasa.gov:/tmp/c_foobar
```

Note that bbscp is just a client-side wrapper for bbftp, thus like bbftp, **you must use the fully-qualified domain name of the target host** (in this case, cfe1.nas.nasa.gov) if you are not within the NAS domain.

- **qstat** (man page available on Pleiades and Columbia))

```
ssh -Aqx -oBatchMode=yes sup.nas.nasa.gov ssh -q cfe1 qstat @pbs1
```

- **rsync** (man page)

```
rsync -e "ssh -Aqx -oBatchMode=yes sup.nas.nasa.gov ssh -q" \
    foobar cfe1:/tmp/c_foobar
```

Note that even if your home directory has been authorized for writes, **rsync transfers to your home directory will fail unless the "-T" or "--temp-dir" option is specified**. This is because rsync uses temporary files starting with "." during transfers, which cannot be written in your home directory. By specifying an alternate temporary directory that is authorized for writes, this problem can be avoided. For example, the following uses /tmp as the temporary directory when files are transferred to the home directory. Make sure that the temporary directory specified has enough space for the files being transferred.

```
rsync -T /tmp -e "ssh -Aqx -oBatchMode=yes sup.nas.nasa.gov ssh -q" \
    foobar cfe1:
```

- **scp** (man page)

    1. Create a file (for example, "supwrap") containing the following:

    ```
    #!/bin/sh
     exec ssh -Aqx -oBatchMode=yes sup.nas.nasa.gov ssh -q $@
    ```

    2. Make the created file executable:

    ```
    chmod 700 supwrap
    ```

    3. Initiate the transfer. For example:

    ```
    scp -S ./supwrap foobar cfe1:/tmp/c_foobar
    ```

- **sftp** (man page)

    1. Create a file (for example, "supwrap") containing the following:

    ```
    #!/bin/sh
    ```

```
        exec ssh -Aqx -oBatchMode=yes sup.nas.nasa.gov ssh -q $@
```

   Note that this file is identical to the one described for scp.

2. Make the created file executable:

```
    chmod 700 supwrap
```

3. Initiate the transfer. For example:

```
    sftp -S ./supwrap cfe1
```

- **test** (man page)

```
ssh -Aqx -oBatchMode=yes sup.nas.nasa.gov ssh -q cfe1 test -f /tmp/c_foobar
```

SUP Troubleshooting

The following error messages may be encountered during SUP usage.

- "WARNING: Your password has expired"

  This message indicates that your current password has expired and must be changed. To change your password, you must log in to an LDAP host (e.g. Lou) through the SFEs and change your LDAP password. This change will be automatically propagated to the SUP within a few minutes.
- "Permission denied (~/.meshrc not found)"

  This message indicates that you have not created a .meshrc file in your home directory on the target host. SUP commands must be authorized for execution on each target host.
- "Permission denied (key expired)"

  SUP keys are only valid for one week from the time of generation. This message indicates that the SUP key used for authentication has expired and is no longer valid. You must generate a new SUP key or use a different SUP key before attempting another operation.
- "Permission denied (publickey,keyboard-interactive)"

  This message indicates that you have not provided the appropriate authentication credentials to the SUP. There may be several causes:

  - If you are generating a SUP key and also receive an "Error copying key..." message, you have not loaded a private key into your SSH agent corresponding to one of the public keys in the authorized_keys file uploaded to sup-key in steps 1-3 of the SUP Key Generation section. You can verify that the correct key is loaded by running "ssh-keygen -l -f uploaded_key_file" and "ssh-agent -l" and checking that the fingerprint of your uploaded key file has been loaded into your SSH agent.

    - ♦ If you have specified -oBatchMode=yes on the command line, a valid SUP key may not been loaded into your SSH agent. There may also be too many keys loaded into your agent. SSH tries each key in the agent sequentially, so a valid key may still fail if it was added to the agent after a number of invalid keys greater than or equal to the login attempt limit. Check the number of keys in the agent using "ssh -l". The agent may be cleared of keys using "ssh-add -D".

    - ♦ If you have specified -oPubkeyAuthentication=no, you have not provided a valid password and/or a valid securID passcode.

- "Permission denied (unauthorized command)"

  This message indicates that you have attempted an operation that is not currently authorized by the SUP. Check that the command line is valid and that the attempted command is one of the authorized commands. Certain options to authorized commands may also be disallowed, but these should never be needed in standard usage scenarios.
- Permission denied during file access (various forms)

  These messages indicate that you attempted to read or write a file for which such access is not allowed. The most common cause is forgetting to authorize directories for writes. Reads and writes of ~/.* are never permitted.

Using GPG to Encrypt Your Data

DRAFT

This article is being reviewed for completeness and technical accuracy.

Introduction

Inter-host file transfer (ex: scp, bbftp, ftp) is better protected when the files are encrypted. GPG (Gnu Privacy Guard) is an Open Source OpenPGP compatible encryption system that we recommend you to use for this purpose. GPG (version 1.4.2) has been installed on Pleiades, Columbia and Lou at /usr/bin/gpg. If you do not have GPG installed on the system(s) that you would like to use for file transfer, please check out the GPG web site. Information, HOWTOs, Guides, FAQs, etc., for GPG can be found at: http://www.gnupg.org

Choosing what cipher to use

We recommend using the cipher AES256, which uses a 256-bit AES key to encrypt the data. Information on AES can be found at:
http://csrc.nist.gov/CryptoToolkit/tkencryption.html

One can set the desired cipher in the following ways:

- add the following line to your ~/.gnupg/gpg.conf

  ```
  cipher-algo AES256
  ```

- add "--cipher-algo AES256" to override the default cipher CAST5 in the command line:

  For any of the following examples in the Simple Examples section, you can add "--cipher-algo AES256" to override the default cipher CAST5 if you chose to not add the "cipher-algo AES256" to your personal gpg.conf file.

Simple Examples

- creating an encrypted file:

  Both commands below are identical. They encrypt the file 'test.out' and produce the encrypted version in 'test.gpg'.

  ```
  gpg --output test.gpg --symmetric test.out
  gpg -o test.gpg -c test.out
  ```

  You will be prompted for a passphrase, which will be used later to decrypt the file.

- decrypting a file:

  The following command decrypts the file "test.gpg" and produces the file "test.out".

  ```
  gpg --output test.out -d test.gpg
  ```

  You will be prompted for the passphrase which you used to encrypt the file.

  If you don't use the "--output" option, output of the command goes to STDOUT.

  If you don't use any flags, it will decrypt to a file without the .gpg suffix. That is,

  ```
  gpg test.gpg
  ```

  results in the decrypted data in a file named "test".

Passphrase Selection

Your passphrase should have lots of entropy. We suggest that you include five words of 5-10 letters in size chosen at random with spaces and/or numbers embedded into words and special characters.

You need to be able to recall the passphrase that was used to encrypt the file.

Factors that Affect Encrypt/Decrypt Speed on NAS HECC Filesystems

We do not recommend using the --armour option for encrypting files that will be transferred to/from NAS HECC systems. This option is mainly to send binary data through email, not scp/bbftp/ftp, etc. The file size tends to be about 33% bigger than without this option and takes about 10-15% longer to encrypt the data.

The level of compression used when encrypting/decrypting affects the time required to complete the operation. There are three options for the compression algorithm: none, zip and zlib.

- *--compress-algo none* or *--compress-algo 0*
- *--compress-algo zip* or *--compress-algo 1*
- *--compress-algo zlib* or *--compress-algo 2*

For example,

```
gpg --output test.gpg --compress-algo zlib --symmetric test.out
```

If your data is not compressible, "--compress-algo 0" (aka none) gives you about a 50% performance increase compared to zip "--compress-algo 1" or zlib "--compress-algo 2".

If your data is highly compressible, choosing zlib or zip will not only give you a 20-50% speed increase, but also reduce the file size by upto 20x. For example, a 517MB highly compressible file was compressed to 30MB on Columbia.

zlib is not compatible with PGP 6.x, but neither is the cipher algorithm AES256. zlib is about 10% faster than zip on Columbia and compresses about 10% better than zip.

Random Benchmark Data

We tested the encryption/decription speed of three different files (1MB, 150MB, 517MB) on Columbia. The file used for the 1MB test was an rpm file, presumably already compressed, since the resultant file sizes for the none/zip/zlib were within 1% of each other. The 150MB file was an ISO, also assumed to be a compressed binary file for the same reasons. The 517MB file is a text file. These runs were performed on a CXFS filesystem and when many other users' jobs were running. Thus, the performance reported here is just for reference, not the best or worst performance you can expect.

- Using AES256 as the cipher algorithm without --armour:

```
1MB file took ~4 secs to encrypt.
150MB took ~35 secs to encrypt.
```

- Using AES256 as the cipher algorithm with --armour:

```
1MB file took ~5.5 secs to encrypt.
150MB took ~40 secs to encrypt.
```

- Using AES256 as the cipher algorithm without --armour, zlib compression:

```
150MB took ~33 secs to encrypt.
decrypt to file: ~28 secs
```

- Using AES256 as the cipher algorithm without --armour, zip compression:

```
150MB took ~36 secs to encrypt.
decrypt to file: ~31 secs
```

- Using AES256 as the cipher algorithm without --armour, no compression:

```
150MB took ~19 secs to encrypt.
decrypt to file: ~25 secs
```

- Using AES256 as the cipher algorithm without --armour, no compression:

```
517MB text file took ~49 secs, resultant filesize ~517MB
decrypt to file: ~75 secs
```

- Using AES256 as the cipher algorithm without --armour, zip compression:

```
517MB text file took ~38 secs, resultant filesize ~33MB
```

```
decrypt to file: ~34 secs
```

- Using AES256 as the cipher algorithm without --armour, zlib compression:

```
517MB text file took ~33 secs, resultant filesize ~30MB
decrypt to file: ~34 secs
```

Checking File Integrity

DRAFT

This article is being reviewed for completeness and technical accuracy.

It is a good practice to check that your data are complete and accurate before and after a file transfer. A common way for checking data integrity is to compute a checksum of the data.

There are multiple algorithms and programs that one can use for computing a checksum. A good checksum algorithm will yield a different result with high probability when the data is accidentally corrupted. If the checksums obtained before and after the transfer match, the data is almost certainly not corrupted.

On NAS HECC systems, the following programs are available:

- *sum*

  computes a checksum using BSD sum or System V sum algorithm; also counts the number of blocks (1KB-block or 512B-block) in a file
- *cksum*

  computes a cyclic redundancy check (CRC) checksum; also counts the number of bytes in a file
- *md5sum*

  computes a 128-bit MD5 checksum which is represented by a 32-character hexadecimal number

For example,

```
%ls -l foo
-rw------- 1 username groupid  67358 Nov 15 11:49 foo

%sum foo
50063    66

%cksum foo
269056887 67358 foo

%md5sum foo
cfe0fc62607e9dc6ea0c231982316b75  foo
```

md5sum is more reliable than *sum* or *cksum* for detecting accidental file curruption, as the chances of accidentally having two files with identical MD5 checksum are extremely small. It is installed by default in most Unix, Linux, and Unix-like operating systems. Users are recommended to compute the *md5sum* of a file before and after the transfer.

The following example shows that the file *foo* is complete and accurate after the transfer based on its md5sum.

```
pfe1% md5sum foo
cfe0fc62607e9dc6ea0c231982316b75  foo

pfe1% scp foo local_username@your_localhost:

your_localhost%md5sum foo
cfe0fc62607e9dc6ea0c231982316b75  foo
```

See **sum, cksum, md5sum** man pages for more information.

File Transfers Tips

When transferring files to NAS systems, there may be some ways to improve your performance without modifying your system (see TPC Performance Tuning for WAN Transfers). Below are some quick and easy techniques to try that may improve your performance rates when transferring files remotely to or from NAS.

- Transfer files from the /nobackup file system, which is often faster than the locally mounted disks.
- If you are using SCP, try adding the "-C" option to enable compression:

  ```
  $ scp -C filename user@remotehost.com:
  ```

  This can sometimes double your performance rates.
- For SCP transfers, use a low process-overhead cipher like *arcfour*:

  ```
  $ scp -carcfour  filename user@remotehost.com:
  ```

  This can increase your rates by 5x, compared to older methods like *3des*.
- If you are transferring from Lou, make sure your file is online first. Use the following DFM commands to determine this:

  ```
  $ dmls -al filename   # show the status of your file.
  $ dmget filename      # retrieve your file from tape prior to transferring.
  ```

  Get the full list of DMF commands.
- Use the bridge nodes to transfer files instead using of the Pleiades and Columbia front ends. These hosts have 10-Gigabit interfaces and more memory to handle both multiple and large file transfers.
- If you are transferring many small files, try using the *tar* command to compress them into a single file prior to transfer. Copying one large file is faster than transferring many small files.
- For files larger than a gigabyte, we recommended using BBFTP software, which can achieve much faster rates than single-stream applications such as SCP or RSYNC.

If you continue experiencing slow transfers and want to work with a network engineer to help improve file transfers, please contact support@nas.nasa.gov.